**LAB #2 – Java Methods**

**Due Date: Week 6**

**Value: 5%**

**Maximum points: 15 points**

**Purpose:** The purpose of this Lab assignment is to:

* Practice the use of instance methods in Java classes
* Practice the use of static methods in Java classes
* Practice with arrays, switches, loops, etc…

**References:** Read Chapter 4, 5, 6, 7.

This material provides the necessary information you need to complete the exercises.

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of classes (instance variable declarations, constructors, getter and setter methods, etc.) | 40% |
| Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results) | 40% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 15% |
| **Total** | 100% |

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

**1 point**

**Overall Instructions**: Be sure to read the following general instructions carefully:

- This lab should be completed individually by all the students.

- You will have to demonstrate your solution in a scheduled lab session and submitting the code **through dropbox link on eCentennial**.

You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228\_sectionNumber\_LabNumber**

Example: **JohSmith\_COMP228\_003\_Lab2**

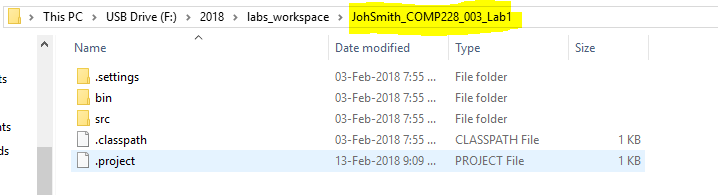
Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

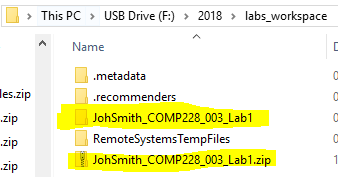
**YourFullName\_COMP228\_sectionNumber\_LabNumber.zip**

Example: **JohSmith\_COMP228\_003\_Lab2.zip**

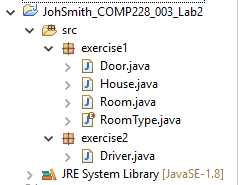
**Project structure looks like that**



**The zipped project folder looks like that:**



sample code structure:



**Exercise 1: (7 points)**

Working with ArrayList, composition, loops, switch, SecureRandom, static variables, enum, method overloading, inner class. Make sure you do not instantiate an object of a class when you call static members of the class.

1. 1 point - Create Room class. See UML below.
2. 2 points - Create Door class. See UML below. Add **final inner class called** Material. This class should contain **constants** for material: “metal”, “wood”, “plastic”.
3. 1 point - Create enum of RoomType. Possible values can be: STUDY, BEDROOM, BATHROOM, etc…
4. 3 points - Create House classClinic . See UML below. A **default constructor should instantiate** the ArrayList of Rooms.

**Exercise 2: (7 points)**

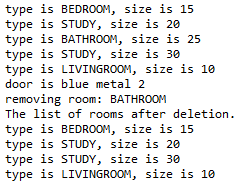
Working with ArrayList, composition, loops, switch, SecureRandom, static variables, enum, method overloading, inner class. Make sure you do not instantiate an object of a class when you call static members of the class.

Create Driver class:

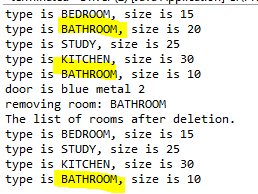
* 1 point - **Sets Door in House.** Make use of inner class Material to set material in the door.
* 2 points - **createRoom**() method should receive all necessary variables to create a new room. Use **Random** **Generator** to generate a random number. Use this random number in combination with **switch** and your **enum** RoomType to randomly set room type to your Room object. Then it should create a new room and return it back. Use this method to create 5 new rooms and store them in the ArrayList in the House object.
* 1 point - **printHouseInfo()** method should receive ArrayList of Rooms and print the all rooms in console view (system.out…). You will have to use some kind of loop structure to iterate though all rooms in the list.
* 1 point -Retrieves Door from House and prints details about the door to the console. Make a use of **printHouseInfo()** methodand **method** **overloading** technique to print door information.
* 2 points – delete on room from array and display back to the user the list of all rooms after deletion. Remove any room you want – example removes bathroom. Remove only one room – the first one found in array. Use key word **break**; to break out of the loop when the first room is found. Make a use of the same method name **printHouseInfo**() using method overloading to be able to display a message that this is the list or rooms after deletion.

**Snapshots of running the code:**

Run 1 – one bathroom – one bathroom removed



Run 2 – two bathrooms – only one is removed



Run 3 – no bathrooms – so no bathroom is removed

